

menu

Menu Component This document outlines the specifications for the `MenuComponent`, an Angular component designed to provide a menu interface for displaying a list of options or actions.

Purpose ■ The `MenuComponent` serves as a user-friendly interface for presenting a list of options or actions to users. It enhances user experience by organizing related items into a structured menu format, facilitating navigation and interaction within an application.

Usage ■ Integrate the `MenuComponent` into your Angular applications to present lists of options or actions in a structured and accessible manner.

Inputs ■ `items` : (Required) An array of objects representing the menu items. Each object should have the following properties:

- `label` : (Required) The label or display text for the menu item.
- `icon` : (Optional) The icon to be displayed alongside the menu item label.
- `action` : (Optional) A callback function to be executed when the menu item is clicked.
- `position` : (Optional) Specifies the position of the menu relative to its parent element. Possible values: 'top-left' 'top-right' 'bottom-left' 'bottom-right'

Events ■ `itemClick` : Fired when a menu item is clicked. Emits the selected menu item object.

Example ■ `<button (click) = " toggleMenu($event) " > Toggle Menu </button> <app-menu *ngIf = " isMenuOpen " [items] = " menuItems " position = " bottom-right " (itemClick) = " onMenuItemClick($event) " > </app-menu>` // Example menu items `menuItems = [{ label : 'Option 1' , icon : 'fa fa-check' } , { label : 'Option 2' , icon : 'fa fa-info' } , { label : 'Option 3' , icon : 'fa fa-cog' , action : this . navigateToSettings }] ;` // Example toggle menu function `isMenuOpen : boolean = false ; toggleMenu (event : MouseEvent) { this . isMenuOpen = ! this . isMenuOpen ; event . stopPropagation () ;` // Prevent click event from propagating to document } // Example menu item click event handler `onMenuItemClick (menuItem : any) { // Handle menu item click event } // Example action for menu item navigateToSettings () { // Navigate to settings page }`

Notes ■ Customize the styling and layout of the menu component to align with the overall design language and branding of your application. Utilize the `position` input to control the placement of the menu relative to its parent element. Implement the `itemClick` event handler to respond to user interactions with the menu items. Ensure that each menu item's action, if provided, performs the intended functionality when executed.